HOW TO DETECT GRAMMATICAL ERRORS IN A TEXT WITHOUT PARSING IT

Eric Steven Atwell
Artificial Intelligence Group
Department of Computer Studies
Leeds University, Leeds LS2 9JT, U.K.
(EARN/BITNET: eric%leeds.ai@ac.uk)

ABSTRACT

The Constituent Likelihood Automatic Word-tagging System (CLAWS) was originally designed for the low-level grammatical analysis of the million-word LOB Corpus of English text samples. CLAWS does not attempt a full parse, but uses a first-order Markov model of language to assign word-class labels to words. CLAWS can be modified to detect grammatical errors, essentially by flagging unlikely word-class transitions in the input text. This may seem to be an intuitively implausible and theoretically inadequate model of natural language syntax, but nevertheless it can successfully pinpoint most grammatical errors in a text. Several modifications to CLAWS have been explored. The resulting system cannot detect *all* errors in typed documents; but then neither do far more complex systems, which attempt a full parse, requiring much greater computation.

Checking Grammar In Texts

A number of researchers have experimented with ways to cope with grammatically ill-formed English input (for example, [Carbonell and Hayes 83], [Charniak 83], [Granger 83], [Hayes and Mouradian 81], [Heidorn et al 82], [Jensen et al 83], [Kwasny and Sondheimer 81], [Weischedel and Black 80], [Weischedel and Sondheimer 83]). However, the majority of these systems are designed for Natural Language interfaces to software systems, and so can assume a restricted vocabulary and syntax; for example, the system discussed by [Fass 83] had a vocabulary of less than 50 words. This may be justifiable for a NL front-end to a computer system such as a Database Query system, since even an artificial subset of English may be more acceptable to users than a formal command or query language. However, for automated text-checking in Word Processing, we cannot reasonably ask the WP user to restrict their English text in this way. This means that WP text-checking systems must be extremely robust, capable of analysing a very wide range of lexical and syntactic constructs. Otherwise, the grammar checker is liable to flag many constructs which are in fact acceptable to humans, but happen not to be included in the system's limited grammar. A system which not only performs syntactic analysis of text, but also pinpoints grammatical errors, must be assessed along two orthogonal scales rather than a single 'accuracy' measure:

RECALL =
"number of words/constructs correctly flagged as errors"
divided by
"total number of 'true' errors that should be flagged"

PRECISION =
"number of words/constructs correctly flagged as errors"
divided by
"total number of words/constructs flagged by the system"

It is easy to optimise one of these performance measures at the expense of the other: flagging (nearly) ALL words in a text will guarantee optimal recall (i.e. (nearly) all actual errors will be flagged) but at a low precision; and conversely, reducing the number of words flagged to nearly zero should raise the precision but lower the recall. The problem is to balance this trade-off to arrive at recall AND precision levels acceptable to WP users. A system which can accept a limited subset of English (and reject (or flag as erroneous) anything else) may have a reasonable recall rate; that is, most of the 'true' errors will probably be included in the rejected text. However, the precision rate is liable to be unacceptable to the WP user: large amounts of the input text will effectively be marked as potentially erroneous, with no indication of where within this text the actual errors lie. One way to deal with this problem is to increase the size and power of the parser and underlying grammar to deal with something nearer the whole gamut of English syntax; this is the approach taken by IBM's EPISTLE project (see [Heidorn et al 82], [Jensen et al 83]). Unfortunately, this can lead to a very large and computationally expensive system: [Heidorn et al 82] reported that the EPISTLE system required a 4Mb virtual machine (although a more efficient implementation under development should require less memory).

The UNIX Writer's Workbench collection of programs (see [Cherry and Macdonald 83], [Cherry et al 83]) is probably the most widely-used system for WP text-checking (and also one of the most widely-used NLP systems overall - see [Atwell 86], [Hubert 85]). This system includes a number of separate programs to check for different types of faults, including misspellings, cliches, and certain stylistic infelicities such as overly long (or short) sentences. However, it lacks a general-purpose grammar checker; the nearest program is a tool to filter out doubled words (as in "I signed the the contract"). Although there is a program PARTS which assigns a part of speech tag to each word in the text (as a precursor to the stylistic analysis programs), this program uses a set of localized heuristic rules to disambiguate words according to context; and these rules are based on the underlying assumption that the input sentences are grammatically well-formed. So, there is no clear way to modify PARTS to flag grammatical errors, unless we introduce a radically different mechanism for disambiguating word-tags according to context.

LOB and CLAWS

One such alternative word-tag disambiguation mechanism was developed for the analysis of the Lancaster-Oslo/Bergen (LOB) Corpus. The LOB Corpus is a million-word collection of English text samples, used for experimentation and inspiration in computational linguistics and related studies (see for example [Leech et al 83a], [Atwell forthcoming b]). CLAWS, the Constituent-Likelihood Automatic Word-tagging System ([Leech et al 83b], [Atwell et al 84]), was developed to annotate the raw text with basic grammatical information, to make it more useful for linguistic research; CLAWS did not attempt a full parse of each sentence, but simply marked each word with a grammatical code from a set of 133 WORDTAGS. The word-tagged LOB Corpus is now available to other researchers (see [Johansson et al 86]).

CLAWS was originally implemented in Pascal, but it is currently being recoded in C and in POPLOG Prolog. CLAWS can deal with Unrestricted English text input including "noisy" or ill-formed sentences, because it is based on Constituent Likelihood Grammar, a novel probabilistic approach to grammatical description and analysis described in [Atwell 83]. A Constituent Likelihood Grammar is used to calculate likelihoods for competing putative analysis; not only does this tell us which is the 'best' analysis, but it also shows how 'good' this analysis is. For assigning word-tags to words, a simple Markovian model can be used instead of a probabilistic rewrite-rule system (such as a probabilistic context-free grammar); this greatly simplifies processing. CLAWS first uses a dictionary, suffixlist and other default routines to assign a set of putative tags to each word; then, for each sequence of ambiguously-tagged words, the likelihood of every possible combination or 'chain' of tags is evaluated, and the best chain is chosen. The likelihood of each chain of tags is evaluated as a product of all the 'links' (tag-pair-likelihoods) in the sequence; tag-pair likelihood is a function of the frequency of that sequence of two tags in a sample of tagged text, compared to the frequency of each of the two tags individually.

An important advantage of this simple Markovian model is that word-tagging is done without parsing: there is no need to work out higher-level constituent-structure trees before assigning unambiguous word-tags to words. Despite its simplicity, this technique is surprisingly robust and successful: CLAWS has been used to analyse a wide variety of Unrestricted English, including extracts form newspapers, novels, diaries, learned journals, E.E.C. regulations, etc., with a consistent accuracy of c96%. Although the system did not have parse trees available in deciding word-classes, only c4% of words in the LOB Corpus had to have their assigned wordtag corrected by manual editing (see [Atwell 81, 82]).

Another important advantage of the simple Markovian model is that it is relatively straightforward to transfer the model from English to other Natural Languages. The basic statistical model remains, only the dictionary and Markovian tag-pair frequency table need to be replaced. We are experimenting with the possibility of (partially) automating even this process - see [Atwell 86a, 86b, forthcoming c], [Atwell and Drakos 87].

The general Constituent Likelihood approach to grammatical analysis, and CLAWS in particular, can be used to analyse text including ill-formed syntax. More importantly, it can also be adapted to flag syntactic errors in texts; unlike other techniques for error-detection, these modifications of CLAWS lead to only limited increases in processing requirements. In fact, various different types of modification are possible, yielding varying degrees of success in error-detection. Several different techniques have been explored.

Error Likelihoods

A very simple adaptation of CLAWS (simple in theory at least) is to augment the tag-pair frequency table with a tag-pair *error likelihood* table. As in the original system, CLAWS uses the tag-pair frequency table and the Constituent Likelihood formulae to find the best word-tag for each word. Having found the best tag for each word, every cooccurring pair of tags in the analysis is re-assessed: the ERROR-LIKELIHOOD of each tag-pair is checked. Error-likelihood is a measure of how frequently a given tag-pair occurs in an error as compared to how frequently it occurs in valid text. For example, if the user types

... *my farther was* ...

CLAWS will yield the word-tag analysis

... *PP$ RBR BEDZ* ...

which means <possessive personal pronoun>, <comparative adverb>, <past singular BE>. This analysis is then passed to the checking module, which uses tag-pair frequency statistics extracted from copious samples of error-full texts. These should show that tag-pairs <PP$ RBR> and <RBR BEDZ> often occur where there is a typing error, and rarely occur in grammatically correct constructs; so an error can be flagged at the corresponding point in the text.

Although the adjustment to the model is theoretically simple, the tag-pair error likelihood frequency figures required could only be gleaned by human analysis of huge amounts of error-full text. Our initial efforts to collect an *Error Corpus* convinced us that this approach was impractical because of the time and effort required to collect the necessary data. In any case, an alternative technique which managed without a separate table of tag-pair error likelihoods turns out to be quite successful.

Low Absolute Likelihoods

This alternative technique involved using CLAWS unmodified to choose the best tag for each word, as before, and then measuring ABSOLUTE LIKELIHOODS of tag-pairs. Instead of a separate tag-pair error likelihood table to assess the grammaticality, the same tag-pair frequency table is used for tag-assignment and error-detection. The tag-pair frequency table gives frequencies for grammatically well-formed text, so the second module simply assumes that if a low-likelihood tag pair occurs in the input text, it indicates a grammatical error. In the example above, tag-pairs <PP$ RBR> and <RBR BEDZ> have low likelihoods (as they occur only rarely in grammatically well-formed text), so an error can be diagnosed.

Figure 1 is a fuller example of this approach to error diagnosis. This shows the analysis of a short text; please note that the text was constructed for illustration purposes

only, and the characters mentioned bear no resemblance to real living people! The text contains many mis-typed words, but these mistakes would not be detected by a conventional spelling-checker, since the error-forms happen to coincide with other legal English words; the only way that these errors can be detected is by noticing that the resultant phrases and clauses are ungrammatical. The grammar-checking program first divides the input text into words. Note that this is not entirely trivial: for example, enclitics such as *I'll*, *won't* are split into two words *I* + *'ll*, *will* + *n't*. The left-hand column in Figure 1 shows the sequence of words in the sample text, one word per line. The second column shows the grammatical tag chosen using the Constituent Likelihood model as best in the given context. The third column shows the absolute likelihood of the chosen grammatical tag; this likelihood is normalised relative to a *threshold*, so that values greater than one constitute "acceptable" grammatical analyses, whereas values less than one are indicative of unacceptably improbable grammar. Whenever the absolute likelihood value falls below this acceptability threshold, the flag *ERROR?* is output in the fourth column, to draw visual attention to the putative error. Thus, for example, the first word in the text, *my*, is tagged *PP$* (possessive personal pronoun), and this tag has a normalised absolute likelihood of over 15, which is acceptable; the second word, *farther*, is tagged *RBR* (comparative adverb), but this time the absolute likelihood is below one (0.264271), so the word is flagged as a putative *ERROR?*

This technique is extremely primitive, yet appears to work fairly well. There is no longer any need to gather error-likelihoods from an Error Corpus. However, the definition of what constitutes a "low" likelihood is not straightforward. On the whole, there is a reasonably clear correlation between words marked *ERROR?* and actual mistakes, so clearly low values *can* be taken as diagnostic of errors, once the question of what constitutes "lowness" has been defined rigorously. In the example, the acceptability level is defined in terms of a simple threshold: likelihoods are normalised so values below 1.000000 are deemed too low to be acceptable. The appropriate normalisation scaling factor was found empirically. Unfortunately, a threshold at this level would mean some minor troughs would not be flagged, e.g. *clever* in *I stole a meat clever,...* (which was tagged JJ (adjective) but should have been the noun *cleaver*) has a normalised likelihood of 4.516465; *tame* in *the gruesome tame of Eroc Attwell...* (which was also tagged JJ (adjective) but should have been the noun *tale*) also has a normalised likelihood of 4.516465; and the phrase *won day* (which should have been *one day*) involves a normalised likelihood of 4.060886 (although this is, strictly speaking, associated with *day* rather than *won*, an error flag would be sufficiently close to the actual error to draw the user's attention to it). However, if we raised the threshold (or alternatively changed the normalisation function so that these normalised likelihoods are below 1.000000), then more words would be flagged, lowering the precision of error diagnosis. In some cases, error diagnosis would be "blurred", since sometimes words immediately before and/or after the error also have low likelihoods; for example, *was* in *my farther was very crawl...* has a likelihood of 1.216545. Worse, some error flags would appear in completely inappropriate places, with no true errors in the immediate context; for example, the exclamation mark at the end of *he won't get away with this!* has a likelihood of 4.185351 and so would probably be flagged as an error if the threshold were raised.

Another way to define a trough would be as a local minimum, that is, a point on where points immediately before and after have higher likelihood values; even a trough with a quite high value is flagged this way so long as surrounding points are even higher. This would catch *clever*, *tame* and *won day* mentioned above. However, strictly speaking several other words not currently flagged in Figure 1 are also local minima, for example *my* in *perhaps my friends would ...* and *if* in *he bald at me if I ...* So, this definition is liable to cause a greater number of 'red herring' valid words to be erroneously flagged as putative mistakes, again leading to a worse precision.

Once an optimal threshold or other computational definition of low likelihood has been chosen, it is a simple matter to amend the output routine to produce output in a simplified format acceptable to Word Processor users, without grammatical tags or likelihood ratings but with putative errors flagged. However, even with an optimal measure of lowness, the success rate is unlikely to be perfect. The model deliberately incorporates only rudimentary knowledge about English: a lexicon of words and their wordtags, and a tag-pair frequency matrix embodying knowledge of tag cooccurrence likelihoods. Certain types of error are unlikely to be detected without some further knowledge. One limited augmentation to this simple model involves the addition of *error tags* to the analysis procedure.

Error-Tags

A rather more sophisticated technique for taking syntactic context into account involves adding ERROR-TAGS to lexical entries. These are the tags of any similar words (where these are different from the word's own tags). In the analysis phase, the system must then choose the best tag (from error-tag(s) and 'own' tag(s)) according to syntactic context, still using the unmodified CLAWS Constituent-Likelihood model. For example, in the sentence *I am very hit*. an error can be diagnosed if the system works out that the tags of input word *hit* (NN, VB, VBD, and VBN - <singular common noun>, <verb infinitive>, <verb past tense>, <verb past participle>) are all much less likely in the given context than JJ (<adjective>), known to be the tag of a similar word (*hot*). So, a rather more sophisticated error-detection system includes knowledge not just about tags of words, but also about what alternative word-classes would be plausible if the input was an error. This information consists in an additional field in lexicon entries: each dictionary entry must hold (i) the word itself, (ii) the word's *own* tags, and (iii) the error-tags associated with the word. For example:

WORD	TAG(S)	ERROR-TAG(S)
...		
form	NN	IN# RI#
...		
hit	NN VB VBD VBN	JJ#
...		
prophecy	NN	VB#
...		

Note that error-tags are marked with # to distinguish them from *own* tags. CLAWS then chooses the best tag for each word as usual. However, in the final output, instead of each word being marked with the chosen word-tag, words associated with an ERROR TAG are flagged as potential errors.

To illustrate why error-tags might help in error diagnosis, notice that *dense* in *I maid several dense in his ...* does not have a below-threshold absolute likelihood, and so is not flagged as a putative error. An error-tag based system could calculate that the best sequence of tags (allowing error-tags) for the word sequence *several dense in his ...* is [AP NNS# IN PP$] (<post-determiner>, <plural common noun>, <preposition>, <possessive personal pronoun>). Since NNS is an error-tag, an error is flagged. However, the simpler absolute likelihood based model does not allow for the option of choosing NNS as the tag for *dense*, and is forced to choose the best of the 'own' tags; this in turn causes a mistagging of *in* as NNU (<abbreviated unit of measurement>, since [JJ NNU] (<adjective> <abbreviated unit of measurement>) is likelier than [JJ IN] (<adjective> <preposition>). Furthermore, [JJ NNU] turns out not to be an exceptionally unusual tag cooccurrence. The point of all this is that, without error-tags, the the system may mistag words immediately before or after error-words, and this mistagging may well distort the absolute likelihoods used for error diagnosis.

This error-tag-based technique was originally proposed and illustrated in [Atwell 83]. The method has been tested with a small test lexicon, but we have yet to build a complete dictionary with error-tags for all words. Adding error tags to a large lexicon is a non-trivial research task; and adding error-tags to the analysis stage increases computation, since there are more tags to choose between for each word. So far, we have not found conclusive evidence that the success rate is increased significantly; this requires further investigation. Also to be more fully investigated is how to take account of other relevant factors in error diagnosis, in addition to error-tags.

Full Cohorts

In theory at least, the Constituent-Likelihood method could be generalised to take account of all relevant contextual factors, not just syntactic bonding. This could be done by generating COHORTS for each input word, and then choosing the cohort-member word which fits the context best. For example, if the sentence *you were very hit* were input, the following cohorts would be generated:

you yew ewe
were where wear
very vary veery
hit hot hut hat
.

(the term "cohort" is adapted from [Marslen-Wilson 85] with a slight modification of meaning). Cohorts of similar words can be discovered from the spelling-check dictionary using the same algorithm employed to suggest corrections for misspellings in current systems; these techniques are fairly well-understood (see, for example, [Yannakoudakis and Fawthrop], [Veronis 87], [Borland 85]). Next, each member of a cohort is assigned a *relative likelihood rating*, taking into account relevant factors including:

i) the degree of similarity to the word actually typed (this measure would be available anyway, as it has to be calculated during cohort generation; the actual word typed gets a similarity factor of 1, and other members of the cohort get appropriate lower weights)

ii) the 'degree of fit' in the given syntactic context (measured as the *syntactic constituent likelihood bond* between the tag(s) of each cohort member and the tag(s) of the words before and after, using the CLAWS constituent likelihood formulae);

iii) the frequency of usage in general English (common words like "you" and "very" get a high weighting factor, rare words like "ewe", "yew", and "veery" get a much lower weighting; word relative frequency figures can be gleaned from statistical studies of large Corpora, such as [Hofland and Johansson 82], [Francis and Kucera 82], [Carroll et al 71]);

iv) if a cohort member occurs in a grammatical idiom or preferred collocation with surrounding words, then its relative weighting is increased (e.g. in the context "fish and ...", *chips* gets a higher collocation weighting than *chops*); collocation preferences can also be elicited from studies of large corpora using techniques such as those of [Sinclair et al 70];

v) domain-dependent lexical preferences should ideally be taken into account, for example in an electronics manual *current* should get a higher domain weighting than *currant*.

All these factors are multiplied (using appropriate weightings) to yield a relative likelihood rating for each member of the cohort. The cohort-member with the highest rating is (probably) the intended word; if the word actually typed is different, an error can be diagnosed, and furthermore a correction can be offered to the user.

Unfortunately, although this approach may seem sensible in theory, in practice it would require a huge R&D effort to gather the statistical information needed to drive such a system, and the resulting model would be computationally complex and expensive. It would be more sensible to try to incorporate only those features which contribute significantly to increased error-detection, and ignore all other factors. This means we must test the existing error-detection system extensively, and analyse the failures to try to discover what additional knowledge would be useful to the system.

Error Corpus

The error-likelihood and full-cohort techniques would appear to give the best error-detection rates, but require vast computations to build a general-purpose system from scratch. The error-tag technique also requires a substantial research effort to build a large general-purpose lexicon. A version of the Constituent Likelihood Automatic Word-tagging System modified to use the ABSOLUTE LIKELIHOOD method of error-detection has been more extensively tested; this system cannot detect *all* grammatical errors, but appears to be quite successful with certain classes of errors. To test alternative prototypes, we are building up an ERROR CORPUS of texts containing errors. The LOB Corpus includes many errors

which appeared in the original published texts; these are marked SIC in the text, and noted in the Manual which comes with the Corpus files, [Johansson et al 78]. The initial Error Corpus consisted in these errors, and it is being added to from other sources (see Acknowledgements below). The errors in the Error Corpus can be (manually) classified according to the kind of processing required for detection (the examples below starts with a LOB line reference number):

A: non-word error-forms, where the error can be found by simple dictionary-lookup; for example,

A21 115 As the news pours in from around the world, beleagured (SIC) Berlin this weekend is a city on a razor's edge.

B: error-forms involving valid English words in an invalid grammatical context, the kind of error the CLAWS-based approach could be expected to detect (these may be due to spelling or typing or grammatical mistakes by the typist, but this is irrelevant here: the classification is according to the type of processing required by the detection program); for example

E18 121 Unlike an oil refinery one cannot grumble much about the fumes, smell and industrial dirt, generally, for little comes out of the chimney except possibly invisible gasses. (SIC)

C: error-forms which are valid English words, but in an abnormal grammatical/semantic context, which a CLAWS-type system would not detect, but which *could* conceivably be caught by a very sophisticated parser; for example, breaking 'long-distance' number agreement rules as in

A15 170 It is, however, reported that the tariff on textiles and cars imported from the Common Market are (SIC) to be reduced by 10 per cent.

D: lexically and syntactically valid error-forms which would require "intelligent" semantic analysis for detection; for example,

P17 189 She did not imagine that he would pay her a visit except in Frank's interest, and when she hurried into the room where her mother was trying in vain to learn the reason of his visit, her first words were of her fiancee. (SIC)

or

K29 35 He had then sown (SIC) her up with a needle, and, after a time she had come back to him cured and able to bear more children.

Collection and detailed analysis of texts for this Error Corpus is still in progress at the time of writing; but one important early impression is that different sources show widely different distributions of error-classes. For example, a sample of 150 errors from three different sources shows the following distribution:

i) Published (and hence manually proofread) text:
A: 52% B: 28% C: 8% D: 12%

ii) essays by 11- and 12-year-old children:
A: 36% B: 38% C: 16% D: 10%

iii) non-native English speakers:
A: 4% B: 48% C: 12% D: 36%

Because of this great variation, precision and recall rates are also liable to vary greatly according to text source. In a production version of the system, the 'unusualness' threshold (or other measure) used to decide when to flag putative errors will be chosen by the user, so that users can optimise precision or recall. It is not clear how this kind of user-customisation could be built into other WP text-checking systems; but it is an obvious side-benefit of a Constituent Likelihood based system.

Conclusions

The figures above indicate that a CLAWS-based grammar-checker would be particularly useful to non-native English speakers; but even for this class of users, precision and recall are imperfect. The CLAWS-based system is inadequate on its own, but should properly be used as one tool amongst many; for example as an augmentation to the Writer's Workbench collection of text-critiquing and proofreading programs, or in conjunction with other English Language Teaching tools such as a computerised ELT dictionary (such as those discussed by [Akkerman et al 85] or [Atwell forthcoming a]. Other systems for dealing with syntactically ill-formed English attempt a full grammatical parse of each input sentence, and in addition require error-recovery routines of varying degrees of sophistication. This involves much more processing than the CLAWS-based system; and yet even these systems fail to diagnose *all* errors in a text. Clearly, the Constituent-Likelihood error-detection technique is ideally suited to applications where fast processing and relatively small computing requirements are of paramount importance, and for users who find imperfect error-detection better than none at all. I freely admit that the system has not yet been comprehensively tested on a wide variety of WP users; as with all AI research systems, a lot af work still has to be done to engineer a generally-acceptable commercial product. We are currently looking for sponsors and collaborators for this research: anyone interested in developing the prototype into a robust system (for example, to be integrated into a WP system) is invited to contact the author!

ACKNOWLEDGEMENTS

This paper was originally produced in 1986 as *Department of Computer Studies Research Report no.212*, Leeds University. I gratefully acknowledge the help of supervisors, colleagues and friends at the Universities of Lancaster and Leeds. The original CLAWS system was developed by Ian Marshall, Roger Garside, Geoffrey Leech and myself at Lancaster University, for a project funded by the Social Science Research Council. Stephen Elliott spent a lot of time building up the Error Corpus and testing variants of the error-detection system, funded by an ICL Research Associateship. Pauline McCrorie and Matthias Wong worked on the POPLOG prolog and C versions of CLAWS. Various other colleagues have also offered advice and encouragement, particularly Geoffrey Sampson, Stuart Roberts, Chris Paice, Lita Taylor, Andrew Beale, Susan

Blackwell, and Barbara Booth.

REFERENCES

Akkerman, Erik, Pieter Masereeuw, and Willem Meijs 1985 *Designing a computerized lexicon for linguistic purposes* Rodopi, Amsterdam

Atwell, Eric Steven 1981 *LOB Corpus Tagging Project: Manual Pre-edit Handbook.* Departments of Computer Studies and Linguistics, University of Lancaster

Atwell, Eric Steven 1982 *LOB Corpus Tagging Project: Manual Postedit Handbook (A mini-grammar of LOB Corpus English, examining the types of error commonly made during automatic (computational) analysis of ordinary written English.)* Departments of Computer Studies and Linguistics, University of Lancaster

Atwell, Eric Steven 1983 "Constituent-Likelihood Grammar" in *Newsletter of the International Computer Archive of Modern English (ICAME NEWS)* 7: 34-67, Norwegian Computing Centre for the Humanities, Bergen University

Atwell, Eric Steven 1986a *Extracting a Natural Language grammar from raw text* Department of Computer Studies Research Report no.208, University of Leeds

Atwell, Eric Steven 1986b, "A parsing expert system which learns from corpus analysis" in Willem Meijs (ed) *Corpus Linguistics and Beyond: Proceedings of the Seventh International Conference on English Language Research on Computerised Corpora, Amsterdam, Netherlands* Rodopi, Amsterdam

Atwell, Eric Steven 1986c "Beyond the micro: advanced software for research and teaching from computer science and artificial intelligence" in Leech, Geoffrey and Candlin, Christopher (eds.) *Computers in English language teaching and research: selected papers from the British Council Symposium on computers in English language education and research, Lancaster, England* 167-183, Longman

Atwell, Eric Steven (forthcoming a) "A lexical database for English learners and users: the Oxford Advanced Learner's Dictionary" to appear in *Proceedings of ICDBHSS87, the 1987 International Conference on DataBases in the Humanities and Social Sciences, Montgomery, Alabama, USA*

Atwell, Eric Steven (forthcoming b) "Transforming a Parsed Corpus into a Corpus Parser", to appear in *Proceedings of the 1987 ICAME 8th International Conference on English Language Research on Computerised Corpora, Helsinki, Finland*

Atwell, Eric Steven (forthcoming c) "An Expert System for the Automatic Discovery of Particles" to appear in *Proceedings of the 1987 International Conference on the Study of Particles, Berlin, East Germany*

Atwell, Eric Steven, Geoffrey Leech and Roger Garside 1984, "Analysis of the LOB Corpus: progress and prospects", in Jan Aarts and Willem Meijs (ed), *Corpus Linguistics; Proceedings of the ICAME Conference on the use of computer corpora in English Language Research, Nijmegen, Netherlands* Rodopi.

E Atwell and N Drakos, "Pattern Recognition Applied to the Acquisition of a Grammatical Classification System from Unrestricted English Text" to appear in the Proceedings of the Association for Computational Linguistics Third European Chapter Conference, 1987 (forthcoming).

Borland International Inc. 1985 *Turbo Lightning: Owner's Handbook* Borland International, Scotts Valley, California USA

Carbonell, Jaime and Philip Hayes 1983 "Recovery strategies for parsing extragrammatical language" in *American Journal of Computational Linguistics* 9(3-4): 123-146

Carroll, John, Peter Davies, and Barry Richman 1971 *The American Heritage word frequency book* Houghton Mifflin / American Heritage

Charniak, Eugene 1983 "A parser with something for everyone" in Margaret King (ed) *Parsing Natural Language* Academic Press, London

Cherry, L, Fox, M, Frase, L, Gingrich, P, Keenan, S, and Macdonald, N 1983 "Computer aids for text analysis" in *Bell Laboratories Records*, May/June: 10-16

Cherry, Lorinda and Macdonald, Nina 1983 "The Writer's WorkBench software" in *BYTE* October: 241-248

Fass, Dan, and Yorick Wilks 1983 "Preference semantics, ill-formedness, and metaphor" in *American Journal of Computational Linguistics* 9(3-4): 178-187

Francis, W Nelson, and Henry Kucera 1982 *Frequency analysis of English usage: lexicon and grammar* Houghton Mifflin

Granger, Richard 1983 "The NOMAD system: expectation-based detection and correction of errors during understanding of syntactically and semantically ill-formed text" in *American Journal of Computational Linguistics* 9(3-4): 188-196

Hayes, Philip J, and G V Mouradian 1981 "Flexible Parsing" in *American Journal of Computational Linguistics* 7(4): 232-242

Heidorn, G E, Jensen, K, Miller, L A, Byrd, R J, and Chodorow, M S, 1982 "The EPISTLE text-critiquing system" in *IBM Systems Journal* 21(3): 305-326

Hofland, Knut and Stig Johansson 1982 *Word frequencies in British and American English* Longman

Hubert, Henry 1985 *Computers and Composition: an annotated bibliography*, English Education 534 Resource Report, University of British Columbia

Jensen, K, Heidorn, G E, Miller, L A, and Ravin, Y 1983 "Parse fitting and prose fixing: getting a hold on ill-formedness" in *American Journal of Computational Linguistics* 9(3-4): 147-160

Johansson, Stig, Geoffrey Leech and Helen Goodluck 1978 *Manual of information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital computers* Department of English, Oslo University

Johansson, Stig, Eric Atwell, Roger Garside, and Geoffrey Leech 1986 *The Tagged LOB Corpus* Norwegian Computing Centre for the Humanities, University of Bergen, Norway.

Kwasny, S and Norman Sondheimer 1981 "Relaxation techniques for parsing grammatically ill-formed input in natural language understanding systems" in *American Journal of Computational Linguistics* 7(2): 99-108

Leech, Geoffrey, Roger Garside, and Eric Steven Atwell 1983a, "Recent developments in the use of computer corpora in English language research" in *Transactions of the Philological Society* 1983: 23-40.

Leech, Geoffrey, Roger Garside, and Eric Steven Atwell 1983b, "The Automatic Grammatical Tagging of the LOB Corpus" in *Newsletter of the International Computer Archive of Modern English (ICAME NEWS)* 7: 13-33, Norwegian Computing Centre for the Humanities, Bergen University

Marslen-Wilson, W D 1985 "Aspects of human speech understanding" in Fallside, Frank and Woods, William (eds.) *Computer speech processing*, Prentice-Hall

Sinclair, John, Jones, S, and Daley, R 1970 *English lexical studies*, Report to OSTI on project C/LP/08; Dept of English, Birmingham University

Veronis, Jean 1987 "Correction of phonographic errors in natural language processing" in Oakman, Robert and Pantonial, Barbara (eds.) *ICCH87: Proceedings of the Eighth International Conference on Computers and the Humanities*, Department of Computer Science, University of South Carolina

Weischedel, Ralph, and John Black 1980 "Responding intelligently to unparsable inputs" in *American Journal of Computational Linguistics* 6(2) 97-109

Weischedel, Ralph, and Norman Sondheimer 1983 "Meta-rules as a basis for processing ill-formed input" in *American Journal of Computational Linguistics* 9(3-4):161-177

Yannakoudakis, E J, and Fawthrop, D 1983 "The rules of spelling errors" in *Information processing and management* 19(2): 87-99

Figure 1. Sample output with low Likelihoods flagged.

my	PP$	15.297639	
farther	RBR	0.264271	ERROR?
was	BEDZ	1.216545	
very	QL	22.137197	
crawl	NN	0.289613	ERROR?
.	.	103.174992	
he	PP3A	90.897396	
bald	JJ	0.271961	ERROR?
at	IN	17.237397	
me	PP1O	29.279452	
if	CS	11.400905	
I	PP1A	71.313009	
dud	JJ	0.271961	ERROR?
anything	PN	0.088535	ERROR?
wrong	JJ	1.682160	
,	,	24.477376	
and	CC	82.096986	
sometimes	RB	29.179920	
he	PP3A	9.921162	
would	MD	64.525545	
bot	JJ	0.220232	ERROR?
and	CC	24.663050	
bit	NN	20.028340	
me	PP1O	0.062710	ERROR?
,	,	18.500350	
until	CS	29.873133	
I	PP1A	71.313009	
was	BEDZ	95.448591	
so	QL	22.137197	
week	NN	0.289613	ERROR?
and	CC	42.917870	
miserable	NN	20.028340	
that	CS	18.439211	
I	PP1A	71.313009	
wanted	VBD	135.815263	
to	TO	28.445266	
due	JJ	0.216826	ERROR?
.	.	21.911547	
finally	RB	36.564715	
,	,	48.403013	
won	VBD	25.409130	
day	NN	4.060886	
,	,	84.114626	
I	PP1A	36.536284	
decided	VBD	135.815263	
to	TO	28.445266	
got	VBD	0.102690	ERROR?
my	PP$	30.396041	
won	VBD	0.099010	ERROR?
back	RP	21.849187	
on	IN	10.259310	
him	PP3O	29.279452	
:	:	3.242075	
I	PP1A	4.764065	
'll	MD	64.525545	
mike	NN	0.123308	ERROR?
him	PP3O	0.062710	ERROR?
pay	VB	10.708766	
;	;	1.396258	
he	PP3A	4.764065	
will	MD	64.525545	
n't	XNOT	95.159151	
get	VB	0.145858	ERROR?
away	RB	29.196041	
with	IN	38.186770	
this	DT	21.792427	
!	!	4.185351	
I	PP1A	90.897396	
stole	VBD	135.815263	
a	AT	39.564677	
meat	NN	191.684559	
clever	JJ	4.516465	
,	,	24.477376	
and	CC	82.096986	
I	PP1A	25.834909	
maid	NN	0.059657	ERROR?
several	AP	2.085110	
dense	JJ	8.725460	
in	NNU	33.948608	
his	PP$	0.306138	ERROR?
hid	VBD	0.099010	ERROR?
with	IN	34.451138	
it	PP3	9.309486	
!	!	11.826017	
it	PP3	62.337141	
must	MD	46.875000	
have	HV	43.513082	
hurt	VB	0.527287	ERROR?
a	AT	45.661755	
lit	VBD	0.037789	ERROR?
!	!	22.778418	
son	NN	9.189478	
the	ATI	4.149936	
gruesome	NN	160.254821	
tame	JJ	4.516465	
of	IN	17.237397	
Eroc	NN	54.835271	
Attwell	NN	26.254356	
appeared	VBN	8.887370	
in	NNU	4.870130	
all	ABN	0.265393	ERROR?
the	ATI	3.499841	
papers	NNS	40.467490	
.	.	70.542872	
perhaps	RB	36.564715	
my	PP$	5.473606	
friends	NNS	44.477694	
would	MD	15.005662	
learnt	VBN	0.237220	ERROR?
to	TO	34.470793	
spell	NN	0.061250	ERROR?
my	PP$	0.545207	ERROR?
name	NN	51.946085	
correctly	JJ	4.516465	
at	IN	17.237397	
last	AP	10.850327	
!	!	3.437432	

Figure 1. Sample output with low Likelihoods flagged.